PENDING CLAIMS AND STATUS THEREOF

1. (currently amended)   A method for processing a low-overhead interrupt, comprising:

providing a list of one or more instructions designated as fast interrupts;

detecting the occurrence of an interrupt condition that requires servicing;

determining that the interrupt condition corresponds to [[a]] **one of the one or more** fast interrupt**s by comparing the interrupt condition to the list**;

loading a first instruction of an interrupt service routine (ISR) into an instruction register for immediate execution;

loading an address of a second instruction within the ISR into a program counter; and

fetching the second ISR instruction while executing the first ISR instruction.

2. (original)   The method according to claim 1, further comprising: storing contents of a prefetch register into a holding register based on the determining.

3. (original)   The method according to claim 1, further comprising: storing a program counter value and a status register value onto a stack based on the determining.

4. (original)   The method according to claim 3, further comprising: executing remaining instructions in the ISR; and returning from the ISR.

5. (original)   The method according to claim 4, wherein the returning comprises:

restoring an instruction from the holding register into the prefetch register.

6. (original)   The method according to claim 5, wherein the returning further comprises popping a program counter value and a status register value from a stack and storing them into respective program counter and status registers.

7. (original)   The method according to claim 6, further comprising executing the instruction restored to the prefetch register and fetching the next instruction for execution based on the status register and the program counter register.

8. (original)   The method according to claim 7, wherein the status register after the returning from the ISR indicates that a repeat instruction is being processed and wherein a loop counter is decremented after executing the instruction restored to the prefetch register.

9. (currently amended)   A processor for handling low-overhead interrupts, comprising:

a list of one or more interrupt instructions designated as fast interrupts;

a first interrupt instruction register storing the first interrupt instruction of a plurality of interrupt service routines;

a holding register;

an interrupt vector register;

a program counter;

an interrupt logic coupled to the registers, upon an interrupt, the interrupt logic determines if the interrupt is [[a]] **in the list of** fast interrupts and, if so, loads a first instruction of an interrupt service routine (ISR) from the first interrupt instruction register into an instruction register for immediate execution and loads an address of a second instruction within the ISR into the program counter based on the interrupt vector register and executing the ISR.

10. (previously presented)   The processor according to claim 9, wherein the interrupt logic further stores the next instruction for regular execution upon return from the interrupt into a holding register.

11. (previously presented)   The processor according to claim 9, wherein the interrupt logic further stores a program counter value and a status register value onto a stack.

12. (original) The processor according to claim 11, wherein the interrupt logic causes a return from the ISR at the end of the ISR.

Response to Non-Final Office Action mailed November 2, 2004

13. (original) The processor according to claim 12, wherein upon the return, the interrupt logic restores an instruction from the holding register into a register for execution.

14. (original) The processor according to claim 13, wherein upon the return the interrupt logic further pops a program counter value and a status register value from a stack and stores them respectively into the program counter and a status registers.

15. (original) The processor according to claim 14, wherein the status register after the returning from the ISR indicates that a repeat instruction is being processed and wherein a loop counter is decremented after executing the instruction restored to the register.